Split by PDF Splitter



2. **Ланге Ю. В.** Низкочастотные методы и средства неразрушающего контроля многослойных конструкций / Ю. В. Ланге. – М. : Машиностроение, 1991. – 272 с.

3. **Нестерук Д. А.** Тепловой неразрушающий контроль воды в сотовых конструкциях / Д. А. Нестерук // АВИА-2011: сб. материалов X Междунар. науч.-техн. конф. – С. 31–34.

4. **Ривин Г. Л.** Ремонт конструкций из полимерных композиционных материалов летательных аппаратов / Г. Л. Ривин. – Ульяновск : УлГТУ, 2000. – 75 с.

5. Справочник по композиционным материалам: в 2 т. / под ред. Дж. Любина. – М. : Машиностроение, 1988. – 584 с.

*Надійшла до редколегії 14.10.2014 р.*

УДК 519.683

**В. И. Усиченко, А. В. Крюков, О. В. Довгун**

*Государственное предприятие «Конструкторское бюро "Южное" имени М. К. Янгеля»*

# МОДЕЛИРОВАНИЕ СЛУЧАЙНЫХ ВЕЛИЧИН С НОРМАЛЬНЫМИ НЕСИММЕТРИЧНЫМИ ПОЛУВЕТВЯМИ НА ОСНОВЕ АЛГОРИТМА МАРЗАГЛИЯ – БРЕЯ

**Показано можливість модифікації відомого в середовищі програмістів алгоритму Марзаглія – Брея для генерування випадкових величин із нормальними напівгілками різного середнього квадратичного відхилення. Зроблено ймовірнісну оцінку зниження швидкодії модифікованого алгоритму через можливість його «холостих» прогонів.**

*Ключові слова:* алгоритм, нормальні несиметричні напівгілки, імовірність, випадкова величина.

**Показана возможность модификации известного в среде программистов алгоритма Марзаглия – Брея с целью генерирования случайных величин с несимметричными нормальными полуветвями. Произведена вероятностная оценка снижения быстродействия алгоритма из-за возможности его «холостых» прогонов.**

*Ключевые слова:* алгоритм, нормальные несимметричные полуветви, вероятность, случайная величина.

**Possibility of updating of algorithm known among programmers as Marzaglija-Bray is shown for the purpose of generating of random variables with asymmetrical normal semibranches. The probability estimation of decrease in performance of algorithm is caused by of possibility of its empty runs is made.**

*Key words:* algorithm, normal asymmetrical semibranches, probability, a random variable.

**Введение.** При моделировании случайных процессов, например с использованием методов Монте-Карло, часто приходится прибегать к генерации значений нормально распределенных случайных величин.

Для указанной генерации хорошо приспособлен алгоритм Марзаглия – Брея, входящий в комплект поставки интегрированной среды разработки *Delphi* последних версий. Он эффективно решает задачу генерации нормально распределенной случайной величины по заданному матожиданию и среднему квадратическому отклонению.

Однако в некоторых задачах, например при расчете гарантийных запасов топлива ракеты-носителя, встречаются величины, значения которых распределены

© В. И. Усиченко, А. В. Крюков, О. В. Довгун, 2014

Split by PDF Splitter



по несиметричному закону, состоящему из двух нормальных полуветвей с общей модой и различными средними квадратическими отклонениями на полуветвях. Генерация значений таких случайных величин с использованием встроенного генератора случайных чисел среды программирования на первый вигляд кажется проблематичной.

Выход, как свидетельствует практика, может заключаться в модификации вышеупомянутого алгоритма Марзаглия – Брея таким образом, чтобы нарушить симметрию ветвей исходной нормальной кривой применительно к конкретному случаю.

**Постановка задачи.** В настоящей статье ставится цель показать, что двукратное применение алгоритма Марзаглия – Брея как вложенного в алгоритм более высокого уровня нарушает симметрию генерируемой им первоначально нормальной кривой, индуцируя различные средние квадратические отклонения на ее ветвях. Однако при анализе описанной ниже модификации исходного алгоритма оказывается, что вновь полученный алгоритм не застрахован от возможных холостых прогонов, способных несколько снизить его быстродействие. Вероятностная оценка снижения быстродействия модифицированного алгоритма также является одной из целей работы.

**Построение алгоритма генерации двух нормальных несимметрических ветвей.** Пусть случайная величина $X$ имеет матожидание $\mu$ и распределение значений, представленное двумя нормальными полуветвями с общей модой и средним квадратическим отклонением $\sigma_1$ для левой и $\sigma_2$ для правой ветви, причем положим для определенности $\sigma_1 < \sigma_2$ (рис. 1, сплошная кривая над областью заливки). Достроим левую ветвь нормальной кривой со средним квадратическим отклонением $\sigma_2$ (пунктир на рис. 1).

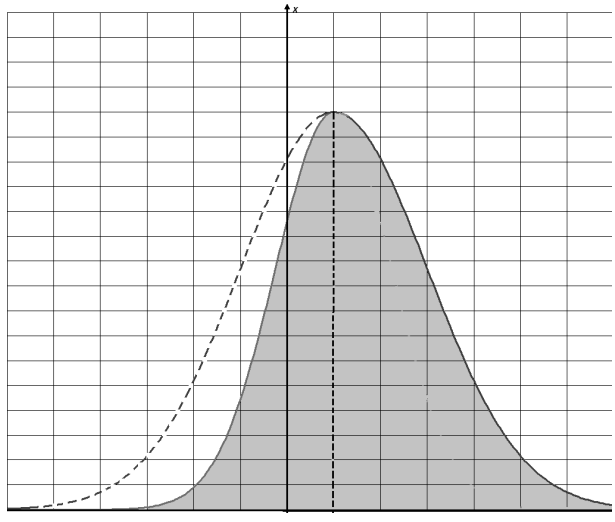


**Рис. 1. Несимметричная кривая значений случайной величины (область заливки) и достроенная полуветвь для модификации алгоритма**

Далее с помощью алгоритма Марзаглия – Брея для матожидания $\mu$ и среднего квадратического отклонения $\sigma_2=\max\{\sigma_1, \sigma_2\}$ генерируем случайное значение $R$ исследуемой величины $X$. Если $R>\mu$, то попадаем под правую ветвь и тем самым получаем одно значение случайной величины $X$ с требуемым распределением. Если же $R<\mu$, то перегенерируем случайное значение величины $X$ с помощью алгоритма Марзаглия – Брея для матожидания $\mu$ и среднего квадратического отклонения $\sigma_1=\min\{\sigma_1, \sigma_2\}$. Получим некоторое случайное значение $R_1$ со средним квадратическим отклонением $\sigma_1$. Если теперь $R_1<\mu$, то $R_1$ соответствует левой ветви несим-

Split by PDF Splitter



метрической кривой и его следует оставить. Если же $R_1$>μ, то попадаем под правую ветвь, где среднее квадратическое отклонение равно $\sigma_2$, и поэтому $R_1$ игнорируем. Затем с помощью алгоритма Марзаглия – Брея для μ и $\sigma_1$ заново генерируем $R_1$ до тех пор, пока не выполнится условие $R_1$<μ. Лишь после этого оставляем сгенерированное случайное значение $R_1$ как соответствующее несимметричному распределению с нормальными полуветвями. Затем возвращаемся в самое начало алгоритма для получения следующего значения случайной величины с требуемым распределением. Описанная процедура представлена в виде блок-схемы модифицированного алгоритма на рис. 2, где блок *Marsaglia_Bray*($\mu,\sigma_{1,2}$) есть подпрограмма (функция) реализации базового алгоритма Марзаглия – Брея для математического ожидания μ и средних квадратических отклонений $\sigma_1$ или $\sigma_2$ соответственно, *ArrRand[i]* – массив сгенерированных значений случайной величины *X* с нормальными несимметричными полуветвями, *i* – счетчик итераций, *N* – число генерируемых значений случайной величины.

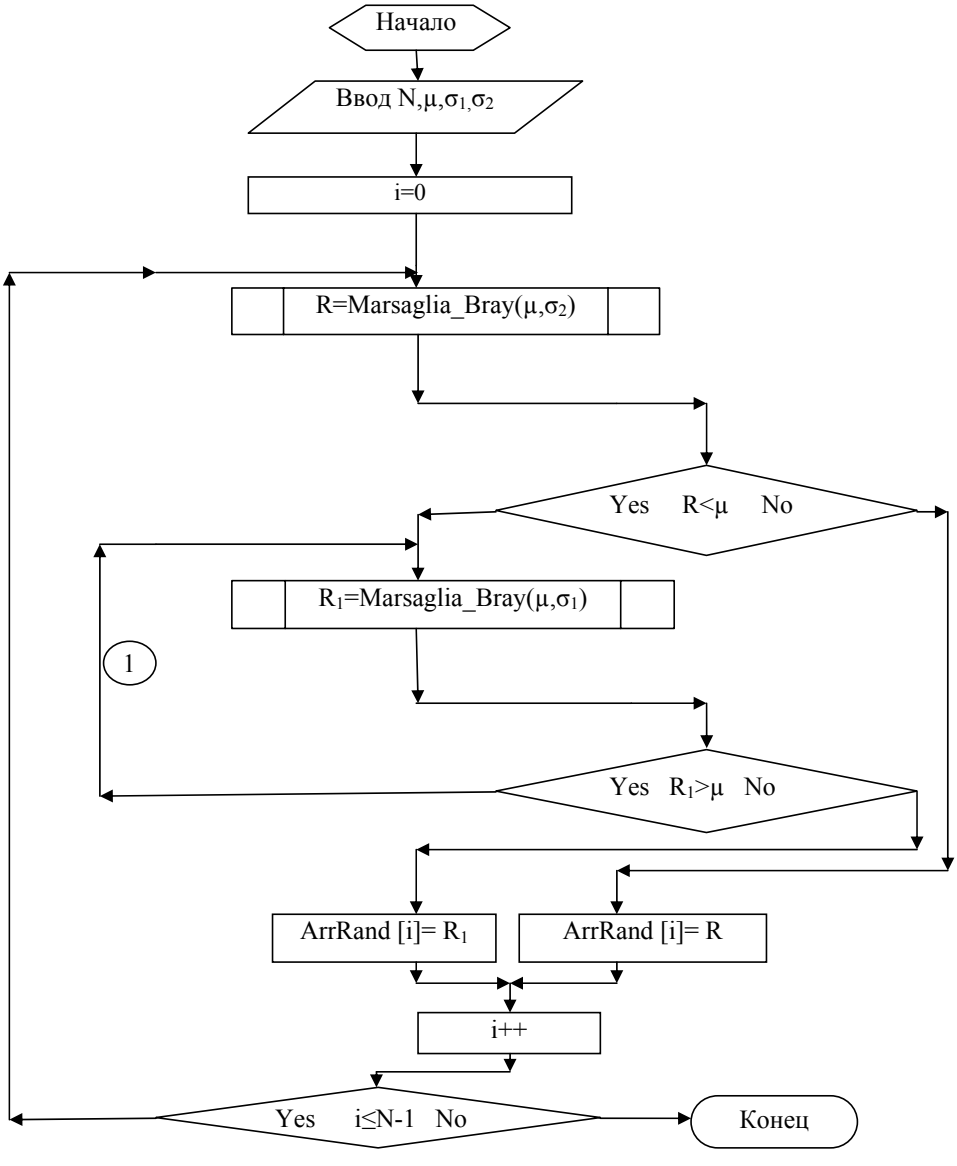


**Рис. 2. Блок-схема модифицированного алгоритма**

Split by PDF Splitter



Из описания принципа работы построенного алгоритма и его блок-схемы (рис. 2) следует, что если сгенерированное для матожидания μ и среднего квадратического отклонения $\sigma_1<\sigma_2$ значение случайной величины $X$ попадает под правую ветвь, где среднее квадратическое отклонение равно $\sigma_2$, то его следует перегенерировать. На блок-схеме это проиллюстрировано рекуррентной ветвью 1. Прогон по этой ветви есть, по сути дела, «холостой» ход алгоритма. Естественно предположить, что число таких «холостых» прогонов будет возрастать с увеличением объема генерируемой совокупности значений. Отсюда возникает естественный вопрос о том, в какой мере такие «холостые» прогоны отразятся на быстродействии построенного алгоритма. Следует также учесть, что приведенная на рис. 2 блок-схема алгоритма допускает, вообще говоря, генерацию случайных значений величины $X$ в неограниченном интервале, тогда как в реальных приложениях случайная величина меняется обычно в некотором конечном интервале $[x_{\min}, x_{\max}]$, ограниченном наименьшим и наибольшим допустимыми значениями моделируемой величины. Это означает, что алгоритм должен игнорировать значения за пределами указанного интервала. Такая возможность предусмотрена в алгоритме, блок-схема которого приведена на рис. 3. Вероятность того, что сгенерированное значение случайной величины $X$ выйдет за левый край интервала $[x_{\min}, x_{\max}]$, равна

$$P(X < x_{\min}) = \frac{1}{\sigma_1\sqrt{2\pi}} \int_{-\infty}^{\mu} e^{-\frac{(x-\mu)^2}{2\sigma_1^2}} dx - \frac{1}{\sigma_1\sqrt{2\pi}} \int_{x_{\min}}^{\mu} e^{-\frac{(x-\mu)^2}{2\sigma_1^2}} dx , \qquad (1)$$

где $x_{\min} = \mu-3\sigma_1$ (правило «трех сигм» для левой полуветви). Первый интеграл в (1), очевидно, равен 0,5. Во втором интеграле производим замену переменной $(x-\mu)/\sigma_1=t$ и получаем:

$$\frac{1}{\sigma_1\sqrt{2\pi}} \int_{\mu-3\sigma_1}^{\mu} e^{-\frac{(x-\mu)^2}{2\sigma_1^2}} dx = \frac{1}{\sqrt{2\pi}} \int_{-3}^{0} e^{-\frac{t^2}{2}} dt = -\frac{1}{\sqrt{2\pi}} \int_{0}^{-3} e^{-\frac{t^2}{2}} dt = -0,5 \cdot \Phi(-3) = 0,5 \cdot \Phi , \qquad (2)$$

где $\Phi(t) = \frac{2}{\sqrt{2\pi}} \int_0^t e^{-\frac{z^2}{2}} dz$ есть интеграл вероятности.

Таким образом окончательно в (1) получаем:

$$P(X < x_{\min}) = 0,5\left(1 - \Phi(3)\right) = 0,00135 .$$

Аналогичный результат получаем и для вероятности $P(X>x_{\max})$ на правой полуветви со средним квадратическим отклонением $\sigma_2$. Сравнивая блок-схемы алгоритмов на рис. 2 и 3, легко заметить, что в случае ограничения интервала генерации случайной величины дополнительно возникают две возвратные ветви (2 и 3 на рис. 3). Их появление, очевидно, увеличивает вероятность Ψ «холостого» хода алгоритма. Поэтому представляется целесообразным оценить ее.

Оценка вероятности перехода алгоритма на возвратные ветви блок-схемы. Для оценки вероятности Ψ «холостого» прогона алгоритма хотя бы по одной из возвратных ветвей представляется целесообразным перейти к схеме событий, обеспечивающей в данном случае наглядную иллюстрацию стохастического характера полученного алгоритма. Введем следующую систему случайных событий (см. рис. 3):

$A_1$ – сгенерированное значение $R$ случайной величины $X$ удовлетворяет условию $R>x_{\max}$;

$A_2$ – сгенерированное значение $R$ случайной величины $X$ удовлетворяет условию $R<\mu$;

Split by PDF Splitter



$A_3$ – сгенерированное значение $R_1$ случайной величины $X$ удовлетворяет условию $R_1 < x_{min}$;

$A_4$ – сгенерированное значение $R_1$ случайной величины $X$ удовлетворяет условию $R_1 > \mu$.

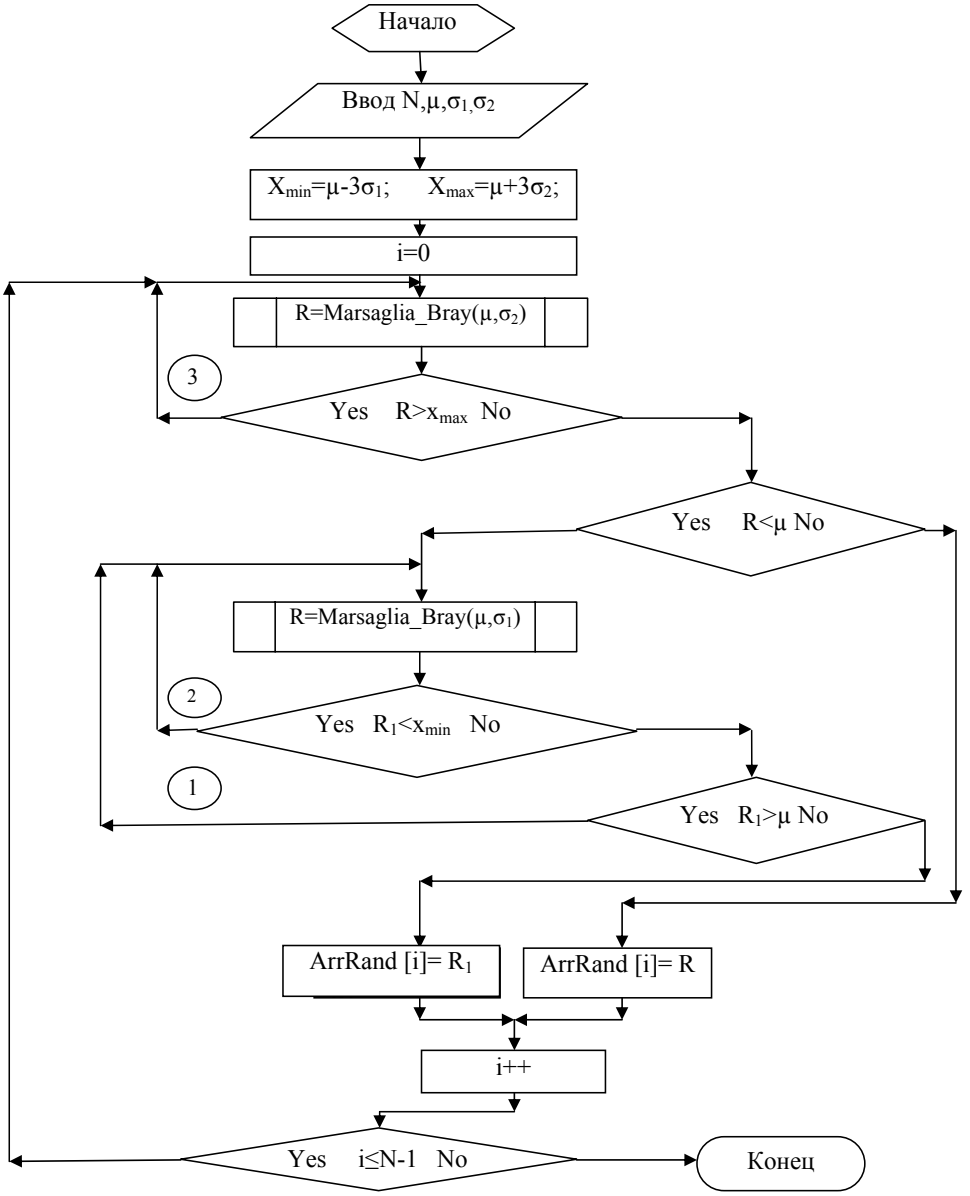


**Рис. 3. Блок-схема модифицированного алгоритма для генерации значений случайной величины на заданном интервале**

Вероятности каждого из этих четырех событий найти несложно. Так как $P(A_1) = P(X > x_{max})$, а $P(A_3) = P(X < x_{min})$, то на основании полученного для (1) результата можем заключить, что $P(A_1) = P(A_3) = 0{,}00135$. Анализ же базового алгоритма Марзаглия – Брея приводит к выводу, что $P(A_2) = P(A_4) = 0{,}5$. Из блок-схемы рис. 3 видим, что управление гарантированно будет хотя бы раз передано как минимум одной из возвратных ветвей при наступлении следующего сложного события $B$:

$$B = A_1 \bigcup \left(\overline{A_1} \cap A_2 \cap A_3\right) \bigcup \left(\overline{A_1} \cap A_2 \cap \overline{A_3} \cap A_4\right) .$$

Split by PDF Splitter



Три события-слагаемые в последнем соотношении являются попарно несовместимыми, поэтому для искомой вероятности $\Psi$ получаем $\Psi = P(B)$ или же

$$\Psi = P(A_1) + P\left(\overline{A_1} \cap A_2 \cap A_3\right) + \\ + P\left(\overline{A_1} \cap A_2 \cap \overline{A_3} \cap A_4\right). \quad (2)$$

На основании теорем сложения и умножения вероятностей и с учетом приведенных выше значений вероятностей для каждого из событий $\{A_i\}$ $i$ = 1,2,3,4 окончательно получим $\Psi$ = 0,75068.

**Выводы.** Как следует из только что произведенной оценки, примерно в 75 процентах случаев построенный нами алгоритм не менее одного раза попадет хотя бы на одну из трех рекуррентных ветвей блок-схемы. Так, например, при генерации совокупности из ста тысяч значений случайной величины хотя бы один возврат алгоритма можно ожидать примерно в 75 000 случаев. Столь частая «пробуксовка» приведенного алгоритма обусловлена его природой, а последняя, в свою очередь, необходимостью генерации именно несимметричных ветвей.

Тем не менее практика свидетельствует, что, несмотря на достаточно высокое полученное значение вероятности $\Psi$, машинное время выполнения алгоритма, например, в не самой быстрой интегрированной среде *Microsoft Visual Basic 5.0 Enterprise Edition* вполне приемлемо. Причина такого явления весьма легко вскрывается. Дело в том, что наиболее критичным по времени выполнения участком алгоритма является подпрограмма (функция) реализации базового алгоритма Марзаглия – Брея, которая с вероятностью 1 выполняется при каждой итерации. К тому же возвратные ветви 1 – 3 алгоритма (см. рис. 3) пустые, а поэтому «пробуксовка» на них слабо сказывается на общем быстродействии алгоритма. Указанные обстоятельства приводят к выводу о том, что в целом полученный алгоритм является алгоритмом вполне приемлемого быстродействия. К тому же практическая проверка свидетельствует (рис. 4), что предложенная структура алгоритма дает весьма качественную совокупность случайных значений с несимметрическими нормальными полуветвями. В этом можно убедиться исходя из вида огибающей для гистограммы в левом нижнем углу рис. 4.

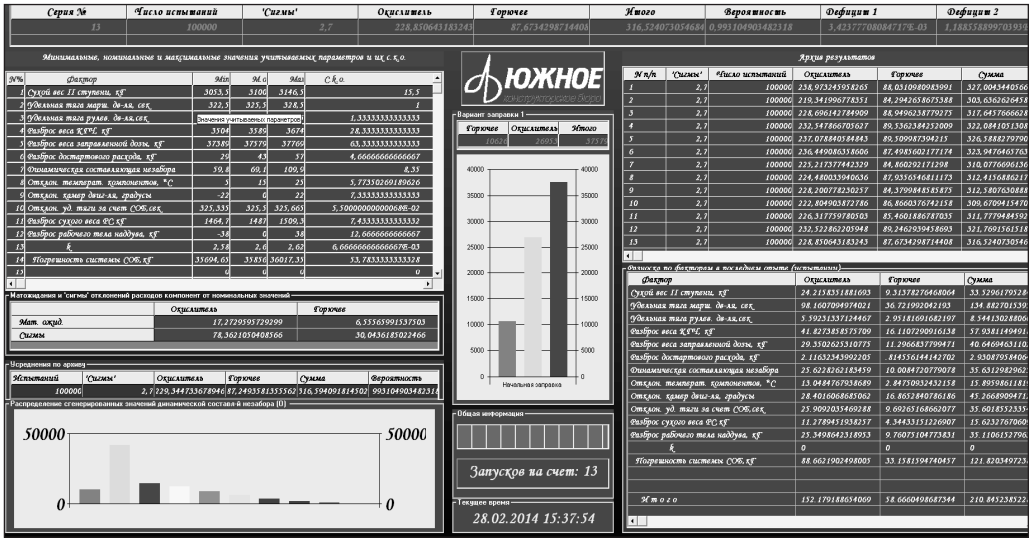


**Рис. 4. Гистограмма сгенерированной случайной величины с нормальными несимметричными ветвями в интерфейсе одной из задач (левый нижний угол)**

Split by PDF Splitter



**Библиографические ссылки**

1. **Демидович Б. П.** Основы вычислительной математики / Б. П. Демидович, И. А. Марон. – М. : Наука, 1970.
2. **Мартин Ф.** Моделирование на вычислительных машинах / Ф. Мартин. – М. : Сов. радио, 1972.

*Надійшла до редколегії 25.09.2014 р.*

УДК 519.683

**В. И. Усиченко, А. В. Крюков**

*Государственное предприятие «Конструкторское бюро "Южное" имени М. К. Янгеля»*

# К ЗАДАЧЕ О РАССТОЯНИЯХ МЕЖДУ ПАРАМИ ЭЛЛИПТИЧЕСКИХ ОРБИТ

**Запропоновано простий алгоритм отримання верхньої оцінки відстані між парою еліптичних орбіт. Наведено аналіз існуючих методів знаходження відстаней між еліптичними орбітами з огляду на зручність їхнього програмування. Головну увагу приділено методу Галле та доцільності його дискретизації. Аналіз його швидкодії для випадку масових обчислень проведено на прикладі конкретної реалізації методу в середовищі Microsoft Visual C++6.0. Коротко висвітлено історичний аспект задачі про відстані між парами орбіт.**

*Ключові слова:* орбіта, верхня оцінка, метод Галле, ітерація, екстремум.

**Предложен простой алгоритм получения верхней оценки расстояния между парой эллиптических орбит. Приведен анализ существующих методов нахождения расстояний между эллиптическими орбитами с точки зрения удобства их программирования. Основное внимание уделено методу Галле и целесообразности его дискретизации. Анализ его быстродействия для случая массовых вычислений выполнен на примере конкретной реализации метода в среде Microsoft Visual C++6.0. Затронут исторический аспект задачи о расстояниях между парами орбит.**

*Ключевые слова:* орбита, верхняя оценка, метод Галле, итерация, экстремум.

**The simple algorithm of deriving of the upper estimation of distance between pair of elliptic orbits is offered. The analysis of existing methods for distance estimation between elliptic orbits from the point of view of convenience of their usage in given programming environment was performed. The basic attention is given for Galle's method and expediency of it's discretization. The performance analysis of Galle's method realization, implemented in Microsoft Visual C++ 6.0 programming environment, was performed. The historical aspect of the task about distance between pair of orbits is briefly covered.**

*Key words:* an orbit, the top estimation, a method of Halle, iteration, an extremum.

**Постановка и формулировка задачи.** Ряд задач космонавтики и астрономии ставят на повестку дня вопросы о сближении и степени совпадения орбит небесних тел. Причем, как известно, задача сближения орбит и задача о степени их совпадения являются принципиально разными, их актуальность для эллиптических орбит особенно очевидна.

Настоящая статья посвящена исключительно задаче сближения эллиптических орбит, на основе которой можно решать задачи отождествления опасных фрагментов небесних тел (как искусственных, так и естественных), выбора безопасных орбит для запуска, а также ряд вопросов, связанных с астрономией ма-

© В. И. Усиченко, А. В. Крюков, 2014